3点系統樹を入力とした系統樹構築の近似アルゴリズムの近似比とその解析 (理論計算機科学の深化 : 新たな計算世界観を求めて)

AUTHOR(S):

前村, 一哉; 小野, 廣隆; 定兼, 邦彦; 山下, 雅史

CITATION:

前村, 一哉 ...[et al]. 3点系統樹を入力とした系統樹構築の近似アルゴリズムの近似比とその解析 (理論計算機科学の深化 : 新たな計算世界観を求めて). 数理解析研究所講究録 2008, 1599: 141-147

ISSUE DATE:

2008-05

URL:

http://hdl.handle.net/2433/81783

数理解析研究所講究録
第1599巻 2008年 141-147

3点系統樹を入力とした系統樹構築の近似アルゴリズムの近似比とその解析

前村 一哉 (Kazuya Maemura) *　　小野 廣隆 (Hirotaka Ono) †
定兼 邦彦 (Kunihiko Sadakane) †　　山下 雅史 (Masafumi Yamashita) †
* 九州大学大学院 システム情報科学府
Graduate School of Electrical Engineering and Computer Science, Kushu University
† 九州大学大学院 システム情報科学研究院
Dept. of Electrical Engineering and Computer Science, Kushu University

1 はじめに

現存する生物種は共通の祖先から進化しており，生物種間には相互に何らかの進化的な類縁関係があると考えられている．系統樹とは，進化の過程において共通祖先から生物がどのように分岐してきたかという生物種間の進化的な関係をグラフの木構造で表現したものである．系統樹は根付き非順序木であり，一意にラベル付けされた葉は生物種に，根は葉となっているすべての生物種の共通祖先に，内部節点は葉や節点の生物同士の祖先に対応し，枝は祖先から子孫へのつながりを表している．また，系統樹は生物種に限らず様々な事柄において，ものの間に存在するであろう相互の由来関係を過去から系譜に沿って体系的に表現・理解する手段として用いられる．

本研究では系統樹は二分木とする．図1は簡単な系統樹の例である [6].

ある種の集合に対して系統樹を構築する一般的な方法は，まず構築したい系統樹の葉(種)の部分集合に対して生物学における研究に基づいて系統樹をつくり，それらを用いて種の集合のすべてを葉とする木を構築するという方法である．しかし，実際には部分集合に対する系統樹においてエラーが生じてしまうために，全ての部分集合の系統樹から全体の木を構築できない可能性があるという問題がある．そこで，種の集合とその部分集合からなる木の集合が与えられたときに，部分集合木における種の進化的な関係をできるだけ多く満たすような系統樹を構築する問題を考える．本研究では3種類の種(葉)からなる3点系統樹と呼ばれる木を入力として，木を構築する．

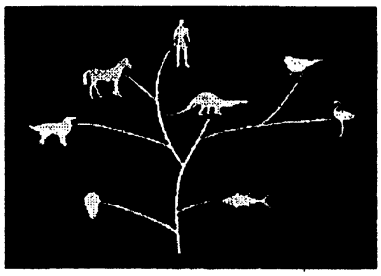

図 1: 系統樹の例

2 準備

本節では，問題の入力となる3点系統樹と3点系統樹を入力とした系統樹構築の問題について述べる．

2.1 3点系統樹

x, y, z を種とすると，3点系統樹は図2のような木で表される．この木は $\{x, y\} < \{x, z\}$ や

$\{x,y\} < \{y,z\}$ のように表記される．このとき $\{x,y\}$ は種 x と y の共通祖先で最も根から遠い祖先，LCA (*Lowest Common Ancestor*) にあたる節点を表している．同様に $\{x,z\}$ は種 x と z の LCA にあたる節点を表している．また，$\{x,y\} < \{x,z\}$ は x と y の LCA は x と z の LCA の正統な子孫であることを示している．つまり，図2は3つの種 x,y,z の進化の過程における関係，まず3つの種の共通祖先から x と y の共通祖先と z に枝分かれし，次に x と y が枝分かれしたという進化の過程における分岐の順序を表している．このように，3点系統樹で表現される3つの種の進化の過程における関係を“進化的関係”と呼ぶことにする．3点系統樹は進化的関係をもつ最小サイズの木であると言える．$\{x,y\} < \{x,z\}$ や $\{x,y\} < \{y,z\}$ は簡潔に $(\{x,y\},z)$ とも表記され，本稿ではこれ以降 $(\{x,y\},z)$ という表記を用いることにする．

2.2 系統樹構築の問題と既存の研究

系統樹構築問題の入力として，2つの集合 S と T が与えられる．S は生物種の集合を，T は S に含まれる3つの種を葉とする3点系統樹の集合を表している; 以降は $|S| = n, |T| = m$ とする．T に含まれる3点系統樹が表す種の進化的関係は，木を構築するための“制約”と考えることができる．本研究で扱う問題は，S と T が与えられたときに T に含まれる3点系統樹が表す制約を満たす木を構築するというものである．本研究では，T に含まれるある要素 T_i における3つの種と構築された木の同じ3つの種の進化的関係が一致するとき，その木は3点系統樹 T_i を“満たす”と表現することにする．

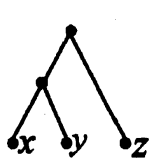

図 2: 3点系統樹の例

系統樹構築に関する既存の研究について述べる．T に含まれるすべての3点系統樹の進化的関係を満たす木を構築できるかという決定問題は，Aho らによって多項式時間で解けることが既に示されている [1]．また，T の3点系統樹ををできる限り多く満たすような木を構築する問題も研究されており，この問題は MRTC (*Maximum Rooted Triplets Consistency*) などと呼ばれ，次のように定義される．この問題は NP 困難であることが証明されており [2,4]，近似アルゴリズムなどの立場から研究されている．

問題 MRTC
入力: S, T
制約: S で一意にラベル付けされた葉をもつ根付き木
コスト: 出力する木が満たす3点系統樹の数
出力: コストを最大化する木

Gasieniec らは [3] で2つの近似アルゴリズム，Split-One-Leaf (SOL) と Min-Cut-Split (MCS) を提案している．SOL は近似比3のアルゴリズムであるが，キャタピラ型の木しか構築しないアルゴリズムである．また，Wu は [5] において近似アルゴリズム Best Pair Merge First (BPMF) を提案している．このアルゴリズムは Wu による計算機実験から SOL, MCS よりも良い近似解を出力するアルゴリズムとされている．しかし，計算機実験のみで近似比などの理論的解析は行われていない．

3 アルゴリズム Best Pair Merge First

本節では，本稿で扱う近似アルゴリズム [5] について説明する．アルゴリズム Best Pair Merge First(以降 BPMF と書く) の詳細を表1に示す．

定義 3.1 関数 $V(t_a)$ を木 t_a が持つ葉の集合とする．

表1の (2-1) で τ からどの2つの木を選ぶかという評価をする基準として，次のような関数 *e_score* を定義している．

Algorithm Best Pair Merge First
(1)$\tau = \{t_i | i \in S,\ t_i$ は節点 i のみの木 $\}$
(2)**while** $|\tau| > 1$ **do**
(2-1) τ から $e\_score(V(t_x), V(t_y))$ の値が最大となる t_x と t_y を選ぶ
(2-2) 共通祖先となる内部節点を加えて t_x と t_y を結合して木を構築，t_x と t_y を構築した木と置換する
endwhile
(3) 構築された木を返す

表 1: アルゴリズム Best Pair Merge First

定義 3.2 関数 *e_score* は次の 3 つの関数から成る．また，*e_score* も以下のように定義する．

- $w(V(t_a), V(t_b)) := \{(\{x,y\},z) \in T | x \in V(t_a), y \in V(t_b), z \in S \backslash \{V(t_a) \cup V(t_b)\}\}$: t_a と t_b を結合して構築された木が満たす 3 点系統樹の集合を表す．
- $p(V(t_a), V(t_b)) := \{(\{x,z\},y),(\{y,z\},x) \in T | x \in V(t_a), y \in V(t_b), z \in S \backslash \{V(t_a) \cup V(t_b)\}\}$: t_a と t_b を結合して構築された木が満たさなくなる 3 点系統樹の集合を表す．
- $t(V(t_a), V(t_b)) := \{(\{x,y\},z) \in T | x \in V(t_a), y \in V(t_b), z \in S\}$: $V(t_a)$ と $V(t_b)$ の要素を $(\{x,y\},z)$ の $\{x,y\}$ にもつ 3 点系統樹の集合を表す．

論文では ***if-penalty*** と ***ratio-type*** という 2 つをパラメータとして，上記 3 つの関数を組み合わせた 6 つの *e_score* が *Wu* によって提案されている．*if-penalty* は 0 か 1 のどちらかが，*ratio-type* は 0,1,2 のいずれかが割り当てられる．2 つのパラメータの値によって，*e_score* の分子と分母の関数が決まる．表 2 に定義された関数 *e_score* の一覧を示す．この表では $|w(V(t_a), V(t_b))|$ を w，$|p(V(t_a), V(t_b))|$ を p，$|t(V(t_a), V(t_b))|$ を t と表記している．

	ratio-type		
if-penalty	0	1	2
0	w	$\frac{w}{w+p}$	$\frac{w}{t}$
1	$w-p$	$\frac{w-p}{w+p}$	$\frac{w-p}{t}$

表 2: 関数 $e\_score(V(t_a), V(t_b))$ の一覧

4 BPMF の近似比

近似アルゴリズムにおいて近似比の解析は主要な研究課題のひとつであるが，[5] では BPMF の近似比について述べられていない．本節では BPMF の近似比について論じる．

4.1 MRTC における近似比

ここで MRTC における近似比について簡単に説明する．

定義 4.1 S と T に対して木 R が満たす 3 点系統樹の数を $Sat(R)$ とする．

R_{Opt}，R_{App} をそれぞれ最適解の木，近似アルゴリズムで構築された木とする．MRTC の近似比は $\max\{\frac{Sat(R_{Opt})}{Sat(R_{App})}\}$ で表すことができる．

4.2 BPMF の近似比 3 の証明

本稿では関数 *e_score* が if-penalty=0，ratio-type=1 のときの近似比を求める．以後，単に BPMF と呼ぶときは *e_score* が if-penalty=0，ratio-type=1 である BPMF とする．まずアルゴリズムや *e_score* についていくつかの命題を証明し，それらの命題を用いて BPMF の近似比 3 を導く．

補題 4.2 任意の反復で τ に含まれる t_x と t_y が *e_score* の値を最大にするとき，$w(V(t_x), V(t_y))$ と $p(V(t_x), V(t_y))$ に含まれる 3 点系統樹は，次以降の反復で計算される *e_score*(w および p) には含まれない．

証明：*e_score* は τ に含まれる異なる 2 つの木を結合して得られる木を評価する．任意の反復で $V(t_x)$ と $V(t_y)$ を引数にしたときに *e_score* の値が最大になると，t_x と t_y が結合して新しい木が構築される．新しく構築された木の葉

の集合は $V(t_x) \cup V(t_y)$ となるため，次の反復で計算される e_score では，引数の片方が $V(t_x) \cup V(t_y)$ となる．そのため，τ に含まれるどの木の葉の集合がもう一方の引数となっても，$e\_score(V(t_x), V(t_y))$ に含まれる 3 点系統樹はその e_score には含まれない．さらに，一度結合した木は分割されることはない．よって，$w(V(t_x), V(t_y))$ と $p(V(t_x), V(t_y))$ に含まれる 3 点系統樹は次以降の反復で計算される e_score には含まれないことが言える． □

1 つ前の各反復までで最大の e_score に含まれる 3 点系統樹の和集合を T' とする．補題 4.2 より T' に含まれる任意の 3 点系統樹では，3 つの葉のうち少なくとも 2 つは τ に含まれるある 1 つの木が持っていることが言えるので，このことより次の系が導かれる．

系 4.3 *$T\backslash T'$ に含まれる任意の 3 点系統樹の 3 つの葉は τ に含まれる 3 つの異なる木が持っている．* □

定義 4.4 *3 点系統樹 $(\{x, y\}, z)$ において，種の組 $\{x, y\}$ を lower-lower-pair，$\{x, z\}$ と $\{y, z\}$ を lower-upper-pair と呼ぶ．*

定義 4.5 以下の 2 つの関数を定義する．

$$LL(k, l; T) := \{(\{k, l\}, z) \in T\}$$
$$LU(k, l; T) := \{(\{k, z\}, l), (\{l, z\}, l) \in T\}$$

補題 4.6 任意の S と T に対して，アルゴリズム中の $T\backslash T'$ が空でない各反復で以下の式が成り立つ．アルゴリズムの初期状態では，T' は空集合とする．

$$\frac{\sum_{k,l \in S} |LL(k, l; T\backslash T')|}{\sum_{k,l \in S} |LL(k, l; T\backslash T')| + \sum_{k,l \in S} |LU(k, l; T\backslash T')|} = \frac{1}{3}$$

証明：任意の 3 点系統樹に対して lower-lower-pair，lower-upper-pair となるような種の組はそれぞれ常に 1 組と 2 組存在する．これより，S に含まれるすべての種の組と $T\backslash T'$ に対して $|LL(k, l; T\backslash T')|$ と $|LU(k, l; T\backslash T')|$ の総和を計算すると，その数の比は 1:2 である．このことより上記の式は成り立つ． □

この補題 4.6 は空でない S と T が与えられたときに，任意の反復で $\frac{|LL(k,l;T\backslash T')|}{|LL(k,l;T\backslash T')|+|LU(k,l;T\backslash T')|}$ が $\frac{1}{3}$ 以上となる種の組が少なくとも 1 組常に存在していることを意味している．

補題 4.7 任意の S と T に対して，アルゴリズム中の $T\backslash T'$ が空でない各反復で以下の式が成り立つ $(t_a, t_b \in \tau)$.

$$\bigcup_{k \in V(t_a), l \in V(t_b)} LL(k, l; T\backslash T') = w(V(t_a), V(t_b))$$
$$\bigcup_{k \in V(t_a), l \in V(t_b)} LU(k, l; T\backslash T') = p(V(t_a), V(t_b))$$

証明：まず 1 つ目の式を証明する．

$\bigcup_{k \in V(t_a), l \in V(t_b)} LL(k, l; T\backslash T')$ に含まれる 3 点系統樹は $V(t_a)$ と $V(t_b)$ の要素が lower-lower-pair となっている 3 点系統樹である．また，系 4.3 より $T\backslash T'$ に含まれる 3 点系統樹がもつ 3 つの葉は，τ に含まれる 3 つの異なる木の葉が持っている．これより，$\bigcup_{k \in V(t_a), l \in V(t_b)} LL(k, l; T\backslash T')$ に含まれる 3 点系統樹 $(\{x, y\}, z)$ の z にあたる種は $S\backslash\{V(t_a) \cup V(t_b)\}$ の要素であると言える．これは $w(V(t_a), V(t_b))$ の定義と同様のことを言っている．よって，$\bigcup_{k \in V(t_a), l \in V(t_b)} LL(k, l; T\backslash T') = w(V(t_a), V(t_b))$ であることが言える．

同様にして $\bigcup_{k \in V(t_a), l \in V(t_b)} LU(k, l; T\backslash T') = p(V(t_a), V(t_b))$ も示すことができる． □

定理 4.8 *BPMF は $m/3$ 以上の T に含まれる 3 点系統樹を満たす木を構築する．*

証明：ある反復で最大の e_score となる 2 つの木を t_x と t_y とする．補題 4.7 から関数 w を LL に p を LP に置き換えることで，この反復の最大の e_score を以下のように変形できる $(\overline{T} = T\backslash T'$ とする$)$.

$$\begin{aligned} & e\_score(V(t_x), V(t_y)) \\ = & \frac{|w(V(t_x), V(t_y))|}{|w(V(t_x), V(t_y))| + |p(V(t_x), V(t_y))|} \\ = & \frac{\sum_{k \in V(t_x), l \in V(t_y)} |LL(k, l; \overline{T})|}{\sum_{k \in V(t_x), l \in V(t_y)} \{|LL(k, l; \overline{T})| + |LU(k, l; \overline{T})|\}} \end{aligned}$$

また，補題 4.6 より上記の式の値は $\frac{1}{3}$ 以上になることが言えるので，$e\_score(V(t_x), V(t_y)) \geq \frac{1}{3}$ となる．

この反復で新しく構築された木は，$T\backslash T'$ に含まれ$V(t_x)$と$V(t_y)$の要素がlower-lower-pairとなっている3点系統樹を満たしている．また$e_score(V(t_x),V(t_y)) \geq 1/3$より，$T\backslash T'$に含まれ$V(t_x)$と$V(t_y)$の要素を両方もつ3点系統樹の総数の$\frac{1}{3}$以上の3点系統樹を満たしている．BPMFはこの過程を繰り返し，$T\backslash T'$が空でない任意の反復での最大のe_scoreの値は$\frac{1}{3}$以上となる．$w(V(t_x),V(t_y))$は各反復で構築される木が満たす3点系統樹の数なので，アルゴリズム全体では満たしている数はTに含まれる3点系統樹の総数の$\frac{1}{3}$以上になっていると言える．よって，出力される木はTに含まれる少なくとも$m/3$の3点系統樹を満たす． □

この定理4.8からアルゴリズムBPMFの近似比を導くことができる．

命題 4.9 *BPMF*の近似比は*3*である．

証明：最適解の木が満たす3点系統樹の数は高々m，定理4.8よりBPMFで構築される木は$m/3$以上の3点系統樹を満たす．R_{BPMF}をBPMFで構築された木とする．

$$\frac{Sat(R_{Opt})}{Sat(R_{BPMF})} \leq \frac{m}{m/3} = 3$$

この式より，BPMFの近似比は3であることが示された． □

しかし，次の命題4.10より上記で求めた近似比3のタイトな例が存在しないことが言える．

命題 4.10 *BPMF*において近似比*3*のタイトな例は存在しない．

証明：背理法を用いて証明する．近似比3のタイトな例が存在すると仮定する．定理4.8より，BPMFで構築される木は$m/3$以上の3点系統樹を満たすため，近似比3のタイトな例では，最適解の木が満たす3点系統樹の数はmとなる．

最適解においても木の構築をBPMFと同様にボトムアップに行うとしても一般性を失わない．どの葉もまだ結合していない初期状態において，最適解の木において2つの子をもち，それらが葉である内部節点でのe_scoreの値を計算する．最適解の木はTに含まれるすべての3点系統樹を満たすため，その値は1となる．

一方，BPMFで構築される近似比3のタイトな例の木は$m/3$の3点系統樹を満たす．また補題4.6から，この近似解においてBPMFでのすべての反復の最大のe_scoreの値は$\frac{1}{3}$となる．つまり，初期状態での最大のe_scoreの値も$\frac{1}{3}$にならなくてはならない．このことは初期状態での最適解の最大のe_scoreの値が1となっていることと矛盾する．よって，BPMFでは近似比3のタイトな例は存在しない． □

4.3 より厳密な近似比

命題4.10より，BPMFにおいて近似比3のタイトな例が存在しないことを証明することができた．そのため，より厳密な近似比を求めることが必要となる．

4.3.1 計算機による比が大きいインスタンスの発見

まず，最適解を求めるアルゴリズムとBPMFを実装したプログラムを用いて，両方の出力される木が満たす3点系統樹の数の比，$\frac{Sat(R_{Opt})}{Sat(R_{BPMF})}$ができるだけ大きくなるような$S$と$T$を探した．その主な手順は次の通りである．$n$と$m$の大きさを決める．0以上$n-1$以下の整数を乱数を用いて発生させ，それらを発生した順に3つずつ組にし，それを3つの葉として3点系統樹をつくる．3点系統樹がm個になるまでこの過程を繰り返してそれをTとし，SとTを入力として$\frac{Sat(R_{Opt})}{Sat(R_{BPMF})}$の値を計算する．$n$と$m$，乱数の種を変えてこの計算を繰り返し行った結果，この比が最大のものとして2となるようなインスタンスが見つかった．

4.3.2 比が2となるインスタンス

計算機を用いて見つかった，比が2となるインスタンスは表3の通りである．表の各列はn，各行はmを表している．各欄は各々のnとmに

	n				
m	5	6	7	8	9
7	6,3	6,3	–	–	–
9	–	–	8,4	8,4	–
11	–	–	–	–	10,5

表 3: 比が 2 となるインスタンスの一覧

対して比が 2 となったインスタンスでの最適解の木が満たす 3 点系統樹の数，BPMF で構築された木が満たす数となっている．また，この過程において見つかった比が 2 となるインスタンスでは，最適解の木はどれもキャタピラ型の木となっている．

$n = 3$ では 3 点系統樹に使用可能な葉が 3 つしかない．そのため，この 3 つの葉を用いた 3 パターンの 3 点系統樹しかつくることができず，明らかに比が 2 となるインスタンスは存在しないことが分かる．また，$n = 4$ では比が 2 となるインスタンスは見つからなかった．

見つかった比が 2 となるインスタンスの中で，$n = 5, m = 7$ の 1 つの例を紹介する．$S = \{0, 1, 2, 3, 4\}$，T は図 3 のような 3 点系統樹の集合，最適解の木は図 4(a)，BPMF で構築される木は図 4(b) となっている．この例では，最適解の木が満たす 3 点系統樹の数は 6，BPMF で構築される木が満たす数は 3 である．

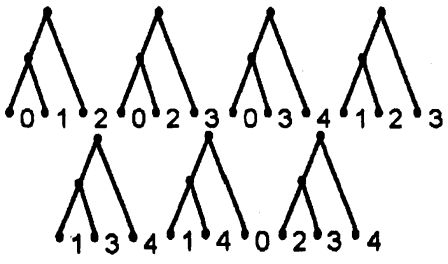

図 3: 比が 2 となるインスタンス ($n = 5, m = 7$ の例) の T に含まれる 3 点系統樹

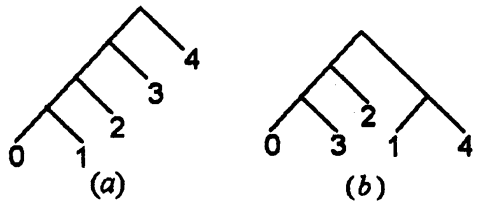

図 4: 比が 2 となるインスタンス ($n = 5, m = 7$) での (a) 最適解の木 (b)BPMF で構築される木の例

4.3.3 近似比 2 の証明は可能か

見つかった比が 2 となるインスタンスでは，BPMF での木の構築における各反復での最大の e_score の中で最小値は $\frac{1}{3}$ である．例えば，$n = 5, m = 7$ の例 (図 3，図 4(b)) では，各反復での最大の e_score の値はそれぞれ $\frac{1}{2}, \frac{1}{2}, \frac{1}{3}$ となっている．この $\frac{1}{3}$ という値は定理 4.8 の証明の過程より，BPMF 中の $T \backslash T' \neq \emptyset$ である任意の反復での最大の e_score の値の下限であることが言える．

また，命題 4.10 において近似比 3 のタイトな例は存在しないことを証明したが，一方で $Sat(R_{BPMF}) = m/3$ となるインスタンスは存在する．例として，ある種の三つ組 $\{x, y, z\}$ に対してつくることができる 3 パターンの 3 点系統樹がすべて T に含まれている場合が挙げられる．この場合，3 パターンの 3 点系統樹のうち必ず 1 つのみを満たすため，T に含まれる 3 点系統樹の総数の $\frac{1}{3}$ を満たしている．

さらに，最適解の木が満たす 3 点系統樹の数の上限は m より下げることはできない．そのため，4.2 節において近似比 3 を証明したような手法で $\frac{Sat(R_{Opt})}{Sat(R_{BPMF})}$ の別の上限を求めること，つまり 2 以上 3 未満となる新たな近似比を導き出すことは難しいと考えられる．

5 まとめ

本稿では Wu が提案した近似アルゴリズム BPMF (if-penalty=0,ratio-type=1) [5] を解析

した．その近似比が3となることを証明し，さらに近似比3となるタイトな例が存在しないことも示した．そのため，より厳密な近似比を求めることが必要となっている．また，計算機によって見つけることができた，近似比が2となるインスタンスを紹介した．

参考文献

[1] A.V.Aho, Y.Sagiv, T.G.Szymanski and J.D.Ullman, "Inferring a tree from lowest common ancestors with an application to the optimizationof relational expressions",*SIAM Journal of Computing*, Vol.10, No.3,pp.405-421, 1981.

[2] D.Bryant, "Building Trees, Hunting for Trees, and Comparing Trees: Theory and Methods in Phylogenetic Analysis,"PhD thesis, University of Canterbury , Christchurch, New Zealand, 1997.

[3] L.Gasieniec, J.Jansson, A.Lingas and A.Östlin, "On the Complexity of Constructing Evolutionary Trees", *Journal of Combinatorial Optimization*, Vol.3, pp.183-197, 1999.

[4] J.Jansson, "On the Complexity of Inferring Rooted Evolurionary Trees," *Proceedings of the Brazirian Symposium on Graphs, Algorithms, and Combinatorics (GRACO 2001), Electronic Notes in Discrete Mathematics*, Vol.7, pp.121-125, Elsevier B.V., 2001.

[5] B.Y.Wu, "Constructing the Maximum Consensus Tree from Rooted Triples", *Journal of Combinatorial Optimization*, Vol.8, No.1, pp.29-39, 2004.

[6] http://www.christiananswers.net/home.html